Parent claim -

US005987601A

United States Patent [19]

Donovan

[11] **Patent Number: 5,987,601**

[45] **Date of Patent: Nov. 16, 1999**

[54] **ZERO OVERHEAD COMPUTER INTERRUPTS WITH TASK SWITCHING**

[75] Inventor: **Brian Donovan**, Wilsonville, Oreg.

[73] Assignee: **Xyron Corporation**, Portland, Oreg.

[21] Appl. No.: **09/023,333**

[22] Filed: **Feb. 13, 1998**

Related U.S. Application Data

[60] Provisional application No. 60/038,729, Feb. 14, 1997.

[51] **Int. Cl.**^{6} **G06F 9/40**

[52] **U.S. Cl.** **712/244**

[58] **Field of Search** 709/1; 712/244

[56] **References Cited**

U.S. PATENT DOCUMENTS

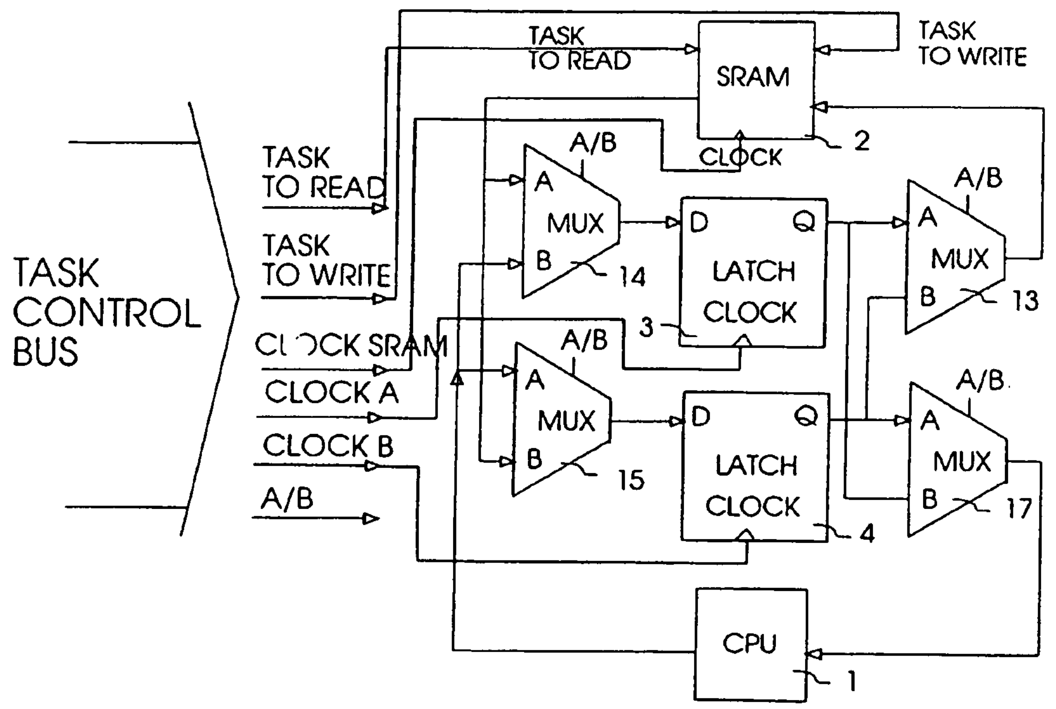
3,226,694	12/1965	Wise	710/264
3,789,365	1/1974	Jen et al.	710/264
4,010,448	3/1977	Bennett et al.	710/269
4,034,349	7/1977	Monaco	710/48
4,047,161	9/1977	Davis	395/678

Primary Examiner—David Y. Eng
Attorney, Agent, or Firm—Chernoff, Vilhauer, McClung & Stenzel, LLP

[57] **ABSTRACT**

The invention constitutes a unique hardware zero overhead interrupt and task change system for the reduction or elimination of interrupt latency and task change processing overhead delays in computer architectures. Without loss of time, the system performs complete task state saving and restoration between one cycle and the next without software intervention. For each Central Processing Unit (1) register, the invention uses one or more auxiliary latches (3, 4) wherein one latch (3, 4) is used as the "running" latch and one of the auxiliary latches is attached to task storage memory. The invention swaps connections between alternate "running" registers and auxiliary registers while transferring other tasks to and from task storage memory (2). The invention provides a task linking system to allow the linking of tasks for the mandatory sequential execution of the linked tasks. Further, the invention includes a priority "impatience" counter system to increase the relative priorities of various tasks as they approach their task deadlines.

1 Claim, 8 Drawing Sheets



WEST

4. (Since our example is from a cold start, there is no task in latch 4. Normally there will be a task there. This task can be written back into the SRAM 2 at the same time the new interrupt task is read out of SRAM 2, by asserting the TASK TO WRITE task destination addresses and then asserting the CLOCK SRAM line.) The new task is loaded into latch 4 at time T2 (FIG. 2). On the same clock edge or a fraction of a cycle later, the A/B control signal is changed to B, causing latch 4 to output and take input from CPU 1, and causing latch 3 to take input and to output to SRAM 2. At this point, if the original task interrupts again, the A/B line is toggled again to switch tasks without requiring a read from SRAM 2.

The above example is the start-up case. Normally, once started, there is a task waiting to run, when a new interrupt comes in. Thus, the waiting task could be "flushed" (thrown away). But, in the preferred embodiment, the waiting task is always run for at least one cycle while the new interrupt task is fetched from the SRAM 2 and the previous running task is saved to SRAM 2. In sequence, when latch 3 is connected to the CPU I and running a task, and while latch 4 is holding the next task scheduled to be run, if a new interrupt of equal or higher priority is detected by the priority selector 19 (FIG. 3), then at T1 (FIG. 2) the A/B line is toggled to B, causing the task data in latch 4 to become attached to the CPU 1, while latch 3 is now connected to the SRAM 2. The new interrupt task address is then put on the TASK TO READ task control bus, and the previous running task address is put on the TASK TO WRITE lines. At T2 (FIG. 2), the previous interrupt task data is clocked into the SRAM 2, the new interrupt task data is clocked into latch 3 and the CPU I will have clocked any data (if it was programmed to) into latch 4. The A/B line is then toggled again to A to run the new interrupt task out of latch 3.

Alternate implementations of the invention are possible. FIG. 4 shows a register bit implemented with 3 latches instead of 2. Here 3-input MUXs 40 & 41 replace the 2-input MUXs in the 2 latch design (FIG. 1). An additional MUX 42 is added to select SRAM 2 or CPU 1. The extra latch allows 3 tasks to have latencies of just I cycle. However, the circuit is larger per bit and somewhat slower because of the extra wiring and capacitance. There are also additional gate delays through MUXs 40 & 41 compared to the simpler MUXs 13 & 17 (FIG. 1). The optimum configuration for a given application depends on the specific characteristics of the integrated circuit manufacturing process used and the application's timing demands.

FIG. 5 shows an implementation of the trace function. In addition to the task switching function described above and shown in FIG. 1, MUX 50 and tri-state switch 51 have been added. Trace SRAM 2*a* is used as either task switching SRAM 2 or as trace SRAM 2*a* by changing the-addressing and trace/task control line to MUX 50. When CPU I stores into latch 3 or 4, if tracing, MUX 50 will pass data to trace SRAM 2*a*. FIG. 6 shows a block diagram of the trace controller. Various test points are selected by MUX 60 to logical testing by test logic 61. If the test logic indicates the situation matches a user test set point, the trace controller 62 begins tracing. The trace controller 62 in its simplest use, outputs sequential trace write addresses while asserting the trace signal. The trace controller can also be designed to trace continuously until an event occurs, then stop tracing, or trace for a few more steps, before stopping. Readout of the trace data is accomplished by asserting the pass control on tri-state 51 (FIG. 5) for each bit to be connected to the trace read bit. This data can be read out a bit at a time, since trace readout is not a time critical task, or it can be grouped together for register output by using a wider bus.

While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example, and not limitation. Thus the breadth and scope of the present invention should not be limited by any of the above described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents. It will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.

The terms and expressions which have been employed in the foregoing specification are used therein as terms of description and not of limitation, and there is no intention, in the use of such terms and expressions, of excluding equivalents of the features shown and described or portions thereof, it being recognized that the scope of the invention is defined and limited only by the claims which follow.

I claim:

1. In a microcomputer having a multiplicity of registers that are selectively multiplexed to communicate with a CPU, and a register set memory for storing a multiplicity of register sets and being dual addressed for reading a first register set simultaneously with writing a second register set, each said register set designated for performing a task, an improvement for permitting the rapid switching between tasks, said improvement comprising:

- a first and a second latch assembly for each bit in said register set, each said latch assembly including:
  - a latch;
  - a first multiplexer having an output connected to the input of said latch, a first input connected to an output of said CPU and a second input connected to an output of said register set memory;
  - a second multiplexer having an input connected to the output of said latch, a first output connected to an input of said CPU and a second output connected to an input of said register set memory, whereby a first register set processed by said CPU may be written into said first latch sets in the same clock cycle as a second register set is read from said register set memory and stored in said second latch sets, and a third register set is read from said second latch sets and written into said register set memory.

* * * * *

WEST